DIGITAL SYSTEM DESIGN LABORATORY

LAB 2

IMPLEMENTATION OF SEQUENTIAL LOGIC CIRCUITS USING VERILOG AND VHDL IN FPGA KIT

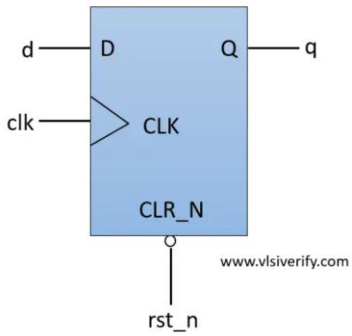
I. LAB OBJECTIVES

This Lab experiments are intended to implement Basic Sequential Circuits in Verilog. Students are require to write test bench to simulate the given example code and Top level module to implement these codes in DE2-115 FPGA Kit.

II. LAB EXPERIMENT EXERCISES

AIM: WRITE VHDL OR VHDL CODES TO SIMULATE AND IMPLEMENT THE FOLLOWING DIGITAL SEQUENTIAL LOGIC CIRCUITS:

1) DFF with Asynchronous Reset using Verilog



Truth Table

reset_n	clk (event)	D	Q(next)	Description
0	X	X	0	When reset is active (low), Q resets immediately
1	↑ (rising edge)	0	0	On clock rising edge, Q follows D = 0
1	↑ (rising edge)	1	1	On clock rising edge, Q follows D = 1
1	0 or 1 (no edge)	X	Q (no change)	When no clock edge, Q holds its previous state

Structural

```
// LINH KI?N: M?t Gated D-Latch v?i Reset không d?ng b?
// (Đây là linh ki?n chúng ta s? dùng d? xây d?ng DFF)
module LAB2 (
    input  wire D,
    input  wire EN,      // Enable (kích ho?t b?ng m?c)
    input  wire RST_N,   // Reset không d?ng b?
    output reg  Q
);

    // Luôn nh?y c?m v?i m?i thay d?i ? d?u vào
    always @(*)
    begin
        if (RST_N == 1'b0)
        begin
            Q = 1'b0; // Reset không d?ng b?
        end
        else if (EN == 1'b1)
        begin
            Q = D;    // Cho d? li?u di qua (transparent)
        end
        // else
        //   Q = Q;   // Gi? giá tr? cu (ng? ý 1 ch?t latch)
    end
endmodule
```

Dataflow

```
module d_latch_dataflow (
    input  wire D,
    input  wire EN,
    input  wire RST_N,
    output reg  Q
);
    wire D_in_to_ff;

    assign D_in_to_ff = (RST_N == 1'b0) ? 1'b0 : D;


endmodule
```

Behavior

```
// LINH KI?N: M?t Gated D-Latch v?i Reset không d?ng b?
// (Đây là linh ki?n chúng ta s? dùng d? xây d?ng DFF)
module gated_latch_with_reset (
    input  wire D,
    input  wire EN,      // Enable (kích ho?t b?ng m?c)
    input  wire RST_N,   // Reset không d?ng b?
    output reg  Q
);

    // Luôn nh?y c?m v?i m?i thay d?i ? d?u vào
    always @(*)
    begin
        if (RST_N == 1'b0)
        begin
            Q = 1'b0; // Reset không d?ng b?
        end
        else if (EN == 1'b1)
        begin
            Q = D;    // Cho d? li?u di qua (transparent)
        end
        // else
        //   Q = Q;   // Gi? giá tr? cu (ng? ý 1 ch?t latch)
    end
endmodule
```

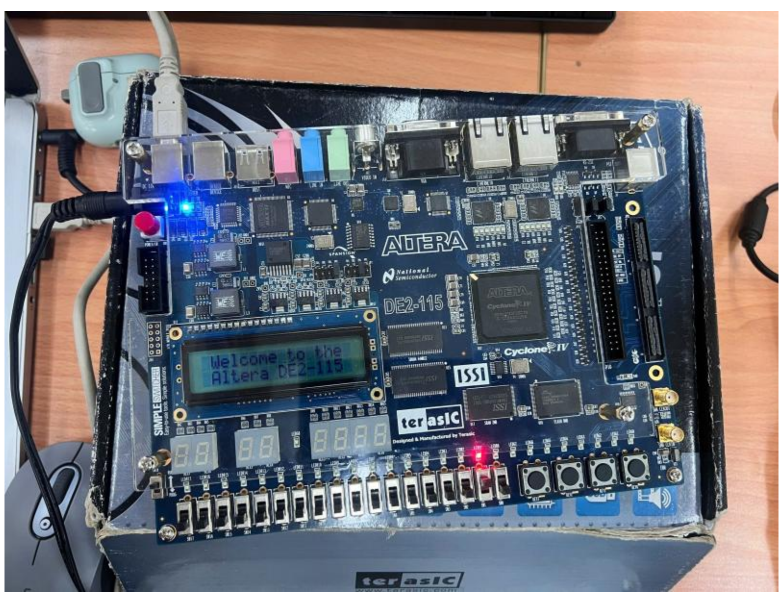
```
/* * MODULE TOP-LEVEL
 * K?t n?i DFF v?i các công t?c và dèn LED
 */
module LAB2 (
    input  wire [17:0] SW,
    output wire [17:0] LEDR
);

    // Dây (wire) d? k?t n?i các tín hi?u
    wire d_in;
    wire clk_in;
    wire rst_n_in;
    wire q_out;

    // --- K?t n?i I/O ---

    // Gán 3 công t?c d?u tiên cho các tín hi?u di?u khi?n DFF
    assign d_in     = SW[0]; // D? li?u D
    assign clk_in   = SW[1]; // Clock (g?t th? công)
    assign rst_n_in = SW[2]; // Reset (m?c th?p)

    // Gán d?u ra Q c?a DFF cho dèn LED d?u tiên
    assign LEDR[0] = q_out;

    // (Tùy ch?n) Gán các công t?c còn l?i cho các dèn LED còn l?i
    // d? d? dàng xem giá tr? d?u vào
    assign LEDR[17:1] = SW[17:1];


    // --- Kh?i t?o (Instantiate) DFF ---

    // Kh?i t?o DFF (s? d?ng module behavioral t? tru?c)
    dff_async_behavioral my_dff (
        .D(d_in),
        .CLK(clk_in),
        .RST_N(rst_n_in),
        .Q(q_out)
    );

endmodule
```

the testbench to simulate the Verilog modules of this circuit

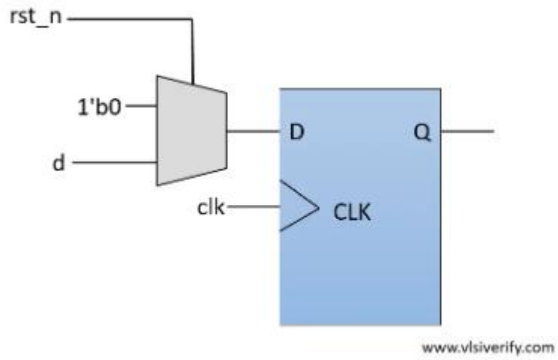
```
 `timescale 1ns/1ps
module Lab2_EX1_tb;
  // Khai báo tín hiệu mô phỏng
  reg [17:0] SW;        // input switch
  wire [17:0] LEDR;     // debug output
  wire [7:0] LEDG;      // output LED (chỉ LEDG[0] dùng)
  // Kết nối DUT (Device Under Test)
  Lab2_EX1 DUT (
    .SW(SW),
    .LEDR(LEDR),
    .LEDG(LEDG)
  );
  // Tạo clock giả trên SW[2]
  initial begin
    SW[2] = 0;
    forever #10 SW[2] = ~SW[2];   // clock chu kỳ 20ns (50MHz)
  end
  // Kịch bản mô phỏng
  initial begin
    // Bắt đầu mô phỏng
    $display("Start simulation...");
    $monitor("Time=%0t | rst_n=%b | d=%b | q=%b", $time, SW[1], SW[0], LEDG[0]);

    // Ban đầu: reset kích hoạt (active-low)
    SW[1] = 0;    // reset = 0 (đang reset)
    SW[0] = 0;    // D = 0
    #50;
    // Bỏ reset
    SW[1] = 1;    // reset = 1 (ngừng reset)
    #20;
    // Thay đổi D và quan sát Q
    SW[0] = 1;    // D = 1
      #40;
      SW[0] = 0;    // D = 0
      #40;
      SW[0] = 1;    // D = 1
      #40;
      // Kích lại reset giữa chừng
      SW[1] = 0;    // reset kích hoạt → Q về 0
      #30;
      SW[1] = 1;    // bỏ reset
      #40;
      // Kết thúc mô phỏng
      $display("End simulation.");
      #100 $stop;
    end
 endmodule
```

The Top-level Verilog Code to implement the Verilog modules of this circuit in DE2-FPGA Kit

```
module Lab2_EX1(SW, LEDR, LEDG);
    input [17:0] SW;
    output [17:0] LEDR;
    output [7:0] LEDG;

    assign LEDR = SW;

    D_flipflop DUT (
        .clk(SW[2]),
        .rst_n(SW[1]),
        .d(SW[0]),
        .q(LEDG[0])
    );
endmodule

module D_flipflop (
    input clk, rst_n,
    input d,
    output reg q
);
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
            q <= 0;
        else
            q <= d;
    end
endmodule
```



2) DFF with Synchronous Reset using VHDL



Synchronous active low reset D flip flop

Step 2: Truth Table

reset	clk (event)	D	Q(next)	Description
1	↑	X	0	On rising edge, reset active → Q = 0
0	↑	0	0	On rising edge, D = 0 → Q = 0
0	↑	1	1	On rising edge, D = 1 → Q = 1
X	0 or 1 (no ↑)	X	Q (no change)	No clock edge → Q holds previous value

VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dff_sync_reset_high is
    Port (
        D   : in  STD_LOGIC;          -- Đ?u vào d? li?u
        CLK : in  STD_LOGIC;          -- Đ?ng h?
        RST : in  STD_LOGIC;          -- Reset d?ng b? (M?c cao)
        Q   : out STD_LOGIC           -- Đ?u ra
    );
end entity dff_sync_reset_high;

architecture Behavioral of dff_sync_reset_high is
begin

    -- Process này CH? nh?y c?m v?i CLK
    process(CLK)
    begin
        -- Ch? th?c thi khi có c?nh lên c?a d?ng h?
        if rising_edge(CLK) then

            -- Ki?m tra reset Đ?NG B? (bên trong if rising_edge)
            if RST = '1' then
                Q <= '0'; -- Uu tiên reset
            else
                Q <= D;   -- Ho?t d?ng DFF bình thu?ng
            end if;

        end if;
    end process;

end architecture Behavioral;
```

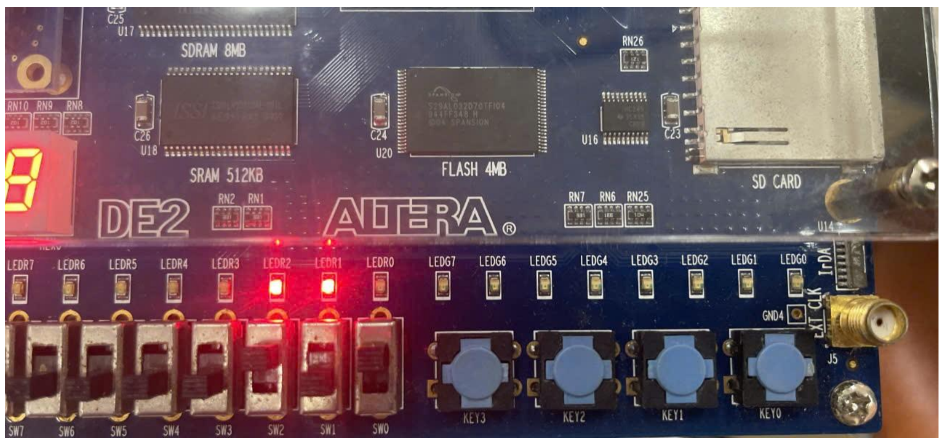
Top Module

```
module Lab2_EX2(SW, LEDR, LEDG);
    input  [17:0] SW;
    output [17:0] LEDR;
    output [7:0]  LEDG;
    assign LEDR = SW;
    D_flipflop DUT (
        .clk  (SW[2]),
        .rst_n(SW[1]),
        .d    (SW[0]),
        .q    (LEDG[2])
    );
endmodule

module D_flipflop (
    input  clk,
    input  rst_n,
    input  d,
    output reg q
);
    always @(posedge clk) begin
        if (!rst_n)
            q <= 1'b0;
        else
            q <= d;
    end
endmodule
```

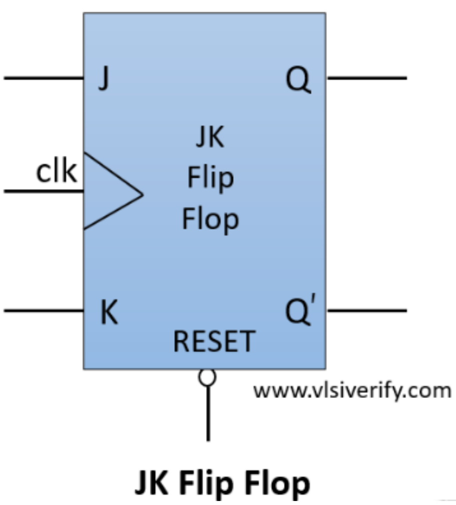
Testbech

```
module Lab2_EX2_tb;
    reg  [17:0] SW;
    wire [17:0] LEDR;
    wire [7:0]  LEDG;
    // Instantiate DUT
    Lab2_EX2 dut (
        .SW   (SW),
        .LEDR (LEDR),
        .LEDG (LEDG)
    );
    // Clock generation (SW[2] is clock)
    initial begin
        SW = 18'b0;
        SW[2] = 0;  // clk
    end
    always #5 SW[2] = ~SW[2];   // 10ns period (100MHz)
    // Test sequence
    initial begin
        $monitor("Time=%0t, rst_n=%b, d=%b, q=%b", $time, SW[1], SW[0],
LEDG[2]);
        // Initial reset = 0
        SW[1] = 0;   // rst_n
        SW[0] = 0;   // d
        #20;
        // Release reset
        SW[1] = 1;
        #20;
        // Test 1: d = 1
```

```
        SW[0] = 1;
        #20;
        // Test 2: d = 0
        SW[0] = 0;
        #20;
        // Test 3: pulse reset
        SW[1] = 0;
        #10;
        SW[1] = 1;
        #20;
        // Finish simulation
        $stop;
    end
endmodule
```



3) JK Flip Flop VHDL



Truth Table

J	K	Q_{n+1}
0	0	Q_n(No Change)
0	1	0
1	0	1
1	1	$\overline{Q_n}$(Toggles)

Testbech

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity Lab2_EX3_JK_tb is
end Lab2_EX_JK_tb;

architecture tb of Lab2_EX3_JK_tb is

  signal SW   : STD_LOGIC_VECTOR(17 downto 0) := (others => '0');
  signal LEDR : STD_LOGIC_VECTOR(17 downto 0);
  signal LEDG : STD_LOGIC_VECTOR(7 downto 0);

  -- DUT declaration
  component Lab2_EX3_JK
    port (
      SW   : in  STD_LOGIC_VECTOR(17 downto 0);
      LEDR : out STD_LOGIC_VECTOR(17 downto 0);
      LEDG : out STD_LOGIC_VECTOR(7 downto 0)
    );
  end component;

begin

  -- Instantiate DUT
  DUT : Lab2_EX3_JK
    port map (
      SW   => SW,
      LEDR => LEDR,
      LEDG => LEDG
    );
  clk_process : process
  begin
    SW(2) <= '0';
    wait for 5 ns;
    SW(2) <= '1';
    wait for 5 ns;
  end process;
  stim_proc : process
  begin
    -- Initial values
    SW(1) <= '0'; -- rst_n
    SW(0) <= '0'; -- J
    SW(3) <= '0'; -- K
    wait for 20 ns;

    -- Release reset
    SW(1) <= '1';
    wait for 20 ns;

    -- Test 1: J=1, K=0 --> Set
    SW(0) <= '1';  -- J
    SW(3) <= '0';  -- K
    wait for 40 ns;

    -- Test 2: J=0, K=1 --> Reset
    SW(0) <= '0';
    SW(3) <= '1';
    wait for 40 ns;

    -- Test 3: J=1, K=1 --> Toggle
```

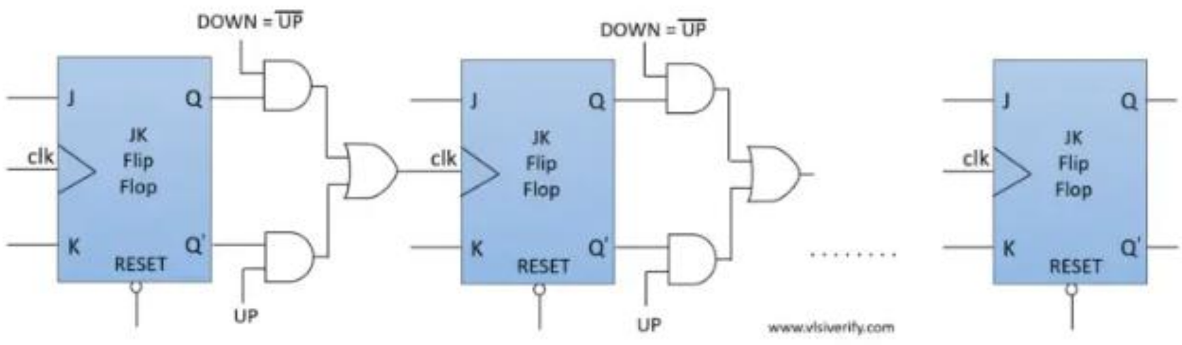
```
        SW(0) <= '1';
        SW(3) <= '1';
        wait for 80 ns;

        -- Test 4: J=0, K=0 --> No change
        SW(0) <= '0';
        SW(3) <= '0';
        wait for 40 ns;

        -- Pulse reset
        SW(1) <= '0';
        wait for 10 ns;
        SW(1) <= '1';
        wait for 40 ns;

        -- End simulation
        wait;
    end process;

end tb;
```

Top Module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Lab2_EX_JK is
    port (
        SW   : in  STD_LOGIC_VECTOR(17 downto 0);
        LEDR : out STD_LOGIC_VECTOR(17 downto 0);
        LEDG : out STD_LOGIC_VECTOR(7 downto 0)
    );
end Lab2_EX_JK;

architecture Behavioral of Lab2_EX_JK is

    component JK_flipflop
        port (
            clk   : in  STD_LOGIC;
            rst_n : in  STD_LOGIC;
            j     : in  STD_LOGIC;
            k     : in  STD_LOGIC;
            q     : out STD_LOGIC;
            q_bar : out STD_LOGIC
        );
    end component;

    signal q_sig, qbar_sig : STD_LOGIC;

begin
    LEDR <= SW;
    LEDG(2) <= q_sig;
    LEDG(3) <= qbar_sig;

    LEDG(7 downto 4) <= (others => '0');
    LEDG(1 downto 0) <= (others => '0');

    -- Instance JK Flip-Flop
    FF_inst : JK_flipflop
```

```
    port map (
        clk   => SW(2),
        rst_n => SW(1),
        j     => SW(0),
        k     => SW(3),
        q     => q_sig,
        q_bar => qbar_sig
    );

end Behavioral;
```

4) Asynchronous Counter 4-bit using VHDL structural modeling



Asynchronous Counter

Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
...	...	...	...	...
15	1	1	1	1
16 (Clear)	0	0	0	0

Testbench

```
module tb;
  reg clk, rst_n;
  reg j, k;
  reg up;
  wire [3:0] q, q_bar;
  asynchronous_counter(clk, rst_n, j, k, up, q, q_bar);

  initial begin
    clk = 0; rst_n = 0;
    up = 1;
    #4; rst_n = 1;
    j = 1; k = 1;
    #80;
    rst_n = 0;
    #4; rst_n = 1; up = 0;
```

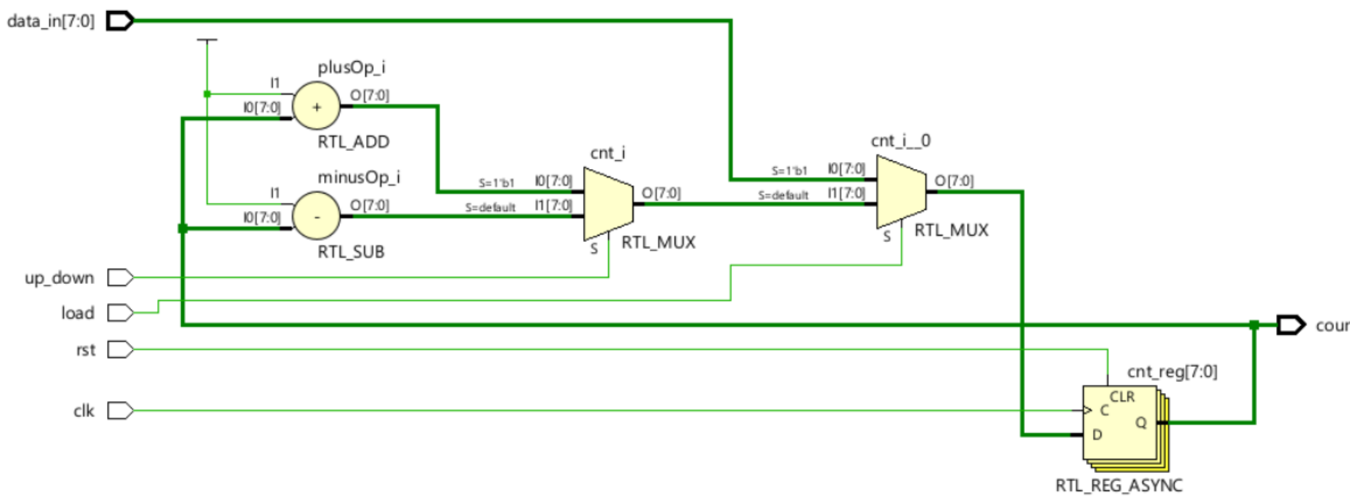
```
    #50;
    $finish;
  end
  always #2 clk = ~clk;

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```

Module

```
module JK_flipflop (
  input clk, rst_n,
  input j,k,
  output reg q, q_bar
  );

  always@(posedge clk or negedge rst_n) begin
    if(!rst_n) q <= 0;
    else begin
      case({j,k})
        2'b00: q <= q;
        2'b01: q <= 1'b0;
        2'b10: q <= 1'b1;
        2'b11: q <= ~q;
      endcase
    end
  end

  assign q_bar = ~q;
endmodule

module updown_selector(input q, q_bar, input up, output nclk);
  assign nclk = up ? q_bar : q;
endmodule

module asynchronous_counter #(parameter SIZE=4)(
  input clk, rst_n,
  input j, k,
  input up,
  output [SIZE-1:0] q, q_bar
);
  wire [SIZE-1:0] nclk;
  genvar g;

  JK_flipflop jk0(clk, rst_n, j, k, q[0], q_bar[0]);

  generate
    for(g = 1; g<SIZE; g++) begin
      updown_selector ud1(q[g-1], q_bar[g-1], up, nclk[g-1]);
      JK_flipflop jk1(nclk[g-1], rst_n, j, k, q[g], q_bar[g]);
    end
  endgenerate
endmodule

//==================== TOP MODULE ====================//
module top_counter(
```

```
  input clk, rst_n,
  input up,
  output [3:0] q
);
  wire [3:0] qb;

  asynchronous_counter #(4) U1(
      .clk(clk),
      .rst_n(rst_n),
      .j(1'b1),
      .k(1'b1),
      .up(up),
      .q(q),
      .q_bar(qb)
  );
endmodule
```

5) Asynchronous Counter 8-bit using VHDL behavior modeling



VHDL Code

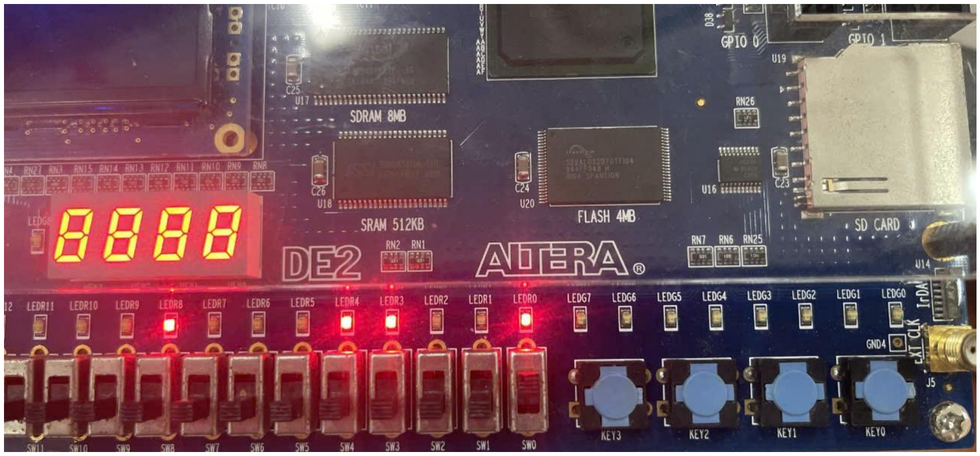
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity up_down_counter_AsyncRst is
   Port (
       clk      : in  STD_LOGIC;
       rst      : in  STD_LOGIC;
       load     : in  STD_LOGIC;
       up_down  : in  STD_LOGIC;
       data_in  : in  STD_LOGIC_VECTOR(7 downto 0);
       count    : out STD_LOGIC_VECTOR(7 downto 0)
   );
end up_down_counter_AsyncRst;

architecture Behavioral of up_down_counter_AsyncRst is
   signal cnt : UNSIGNED(7 downto 0);
begin
   process(rst, clk)
   begin
```

```vhdl
        if rst = '1' then
            cnt <= (others => '0');
        elsif rising_edge(clk) then
            if load = '1' then
                cnt <= UNSIGNED(data_in);
            elsif up_down = '1' then
                cnt <= cnt + 1;
            else
                cnt <= cnt - 1;
            end if;
        end if;
    end process;

    count <= STD_LOGIC_VECTOR(cnt);
end Behavioral;
```

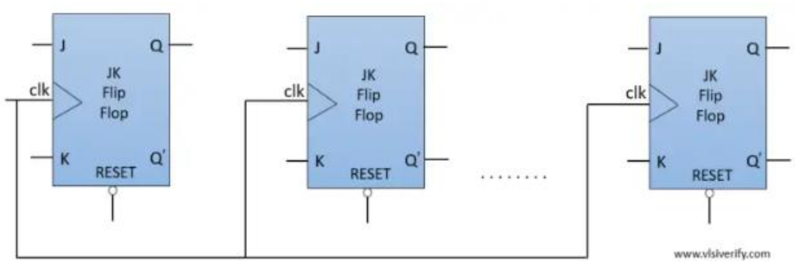
Testbench

```vhdl
-- File: tb_up_down_counter_AsyncRst.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity tb_up_down_counter_AsyncRst is
end tb_up_down_counter_AsyncRst;

architecture Behavioral of tb_up_down_counter_AsyncRst is
    -- Component declaration for the Unit Under Test (UUT)
    component up_down_counter_AsyncRst
        Port (
            clk      : in  STD_LOGIC;
            rst      : in  STD_LOGIC;
            load     : in  STD_LOGIC;
            up_down  : in  STD_LOGIC;
            data_in  : in  STD_LOGIC_VECTOR(7 downto 0);
            count    : out STD_LOGIC_VECTOR(7 downto 0)
        );
    end component;

    -- Signals to connect to UUT
    signal clk_sig     : STD_LOGIC := '0';
    signal rst_sig     : STD_LOGIC := '0';
    signal load_sig    : STD_LOGIC := '0';
    signal up_down_sig : STD_LOGIC := '1';
    signal data_in_sig : STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
    signal count_sig   : STD_LOGIC_VECTOR(7 downto 0);
begin

    -- Instantiate UUT
    UUT: up_down_counter_AsyncRst
        Port map (
            clk     => clk_sig,
            rst     => rst_sig,
            load    => load_sig,
            up_down => up_down_sig,
            data_in => data_in_sig,
            count   => count_sig
        );
```

```
    -- Clock generation: 100 MHz (10 ns period)
    clk_process: process
    begin
        while true loop
            clk_sig <= '0';
            wait for 5 ns;
            clk_sig <= '1';
            wait for 5 ns;
        end loop;
    end process;

    -- Stimulus process
    stim_proc: process
    begin
        -- Apply asynchronous reset
        rst_sig <= '1';
        wait for 20 ns;
        rst_sig <= '0';
        wait for 20 ns;

        -- Synchronous load test
        data_in_sig <= x"AA";  -- load 0xAA
        load_sig <= '1';
        wait for 10 ns;        -- on next rising edge, count = 0xAA
        load_sig <= '0';
        wait for 20 ns;

        -- Count up for 5 cycles
        up_down_sig <= '1';
        wait for 50 ns;

        -- Count down for 5 cycles
        up_down_sig <= '0';
        wait for 50 ns;

        -- Load new value and count up again
        data_in_sig <= x"0F";
        load_sig <= '1';
        wait for 10 ns;
        load_sig <= '0';
        up_down_sig <= '1';
        wait for 40 ns;

        -- Finish simulation
        wait;
    end process;

end Behavioral;
```



6) Synchronous Counter 4-bit using Verilog structural modeling



Synchronous Counter

Clock Pulse	QD	QC	QB	QA	Decimal Equivalen
Initially	0	0	0	0	0
1st Falling Edge	0	0	0	1	1
2nd Falling Edge	0	0	1	0	2
3rd Falling Edge	0	0	1	1	3
4th Falling Edge	0	1	0	0	4
5th Falling Edge	0	1	0	1	5
6th Falling Edge	0	1	1	0	6
7th Falling Edge	0	1	1	1	7
8th Falling Edge	1	0	0	0	8
9th Falling Edge	1	0	0	1	9
10th Falling Edge	1	0	1	0	10
11th Falling Edge	1	0	1	1	11
12th Falling Edge	1	1	0	0	12
13th Falling Edge	1	1	0	1	13
14th Falling Edge	1	1	1	0	14
15th Falling Edge	1	1	1	1	15
16th Falling Edge	0	0	0	0	0

Testbench

```
`timescale 1ns / 1ps

module tb_sync_counter_4bit;

    // 1. Tao tin hieu
    reg  tb_clk;
    reg  tb_rst_n;
    wire [3:0] tb_q_out;

    localparam T_CLK = 10; // Chu ky clock 10ns

    // 2. Khoi tao DUT
    sync_counter_4bit_struc uut (
        .CLK(tb_clk),
        .RST_N(tb_rst_n),
        .Q_out(tb_q_out)
    );

    // 3. Tao Clock
    initial begin
        tb_clk = 0;
        forever #(T_CLK / 2) tb_clk = ~tb_clk;
    end

    // 4. Tao Stimulus
    initial begin
        $display("Time  | RST_N | Q_out");
        $monitor("%0t | %b     | %b (%d)", $time, tb_rst_n, tb_q_out, tb_q_out);

        // Test 1: Kiem tra Reset
        tb_rst_n = 1'b0; // Kich hoat Reset
        # (T_CLK * 2);

        // Test 2: Nha Reset va dem
        tb_rst_n = 1'b1; // Nha Reset

        // Cho dem 20 chu ky
        # (T_CLK * 20);

        // Test 3: Reset lai
        tb_rst_n = 1'b0;
        # (T_CLK * 2);

        $finish;
    end

endmodule
```

Top Module

```
module LAB2 (
    input  wire CLOCK_50,
    input  wire [17:0] SW,
    output wire [17:0] LEDR
);
```

```
    // 1. Tao tin hieu noi bo
    wire clk_1Hz_signal;
    wire reset_n_signal;
    wire [3:0] q_4bit_out;

    // 2. Ket noi cong tac (SW)
    assign reset_n_signal = SW[0]; // SW[0] la Reset muc thap (Active-Low)

    // 3. Khoi tao Clock Divider
    clock_divider clk_div_inst (
        .clk_in(CLOCK_50),
        .clk_out(clk_1Hz_signal)
    );

    // 4. Khoi tao Bo dem Dong Bo 4-bit
    sync_counter_4bit_struc uut (
        .CLK(clk_1Hz_signal),
        .RST_N(reset_n_signal),
        .Q_out(q_4bit_out)
    );

    // 5. Ket noi den LED
    assign LEDR[3:0] = q_4bit_out;     // 4 bit dem
    assign LEDR[4]   = clk_1Hz_signal; // Clock 1Hz nhap nhay
    assign LEDR[17:5] = SW[17:5];

endmodule
```

Module Syschronous Counter 4-bit

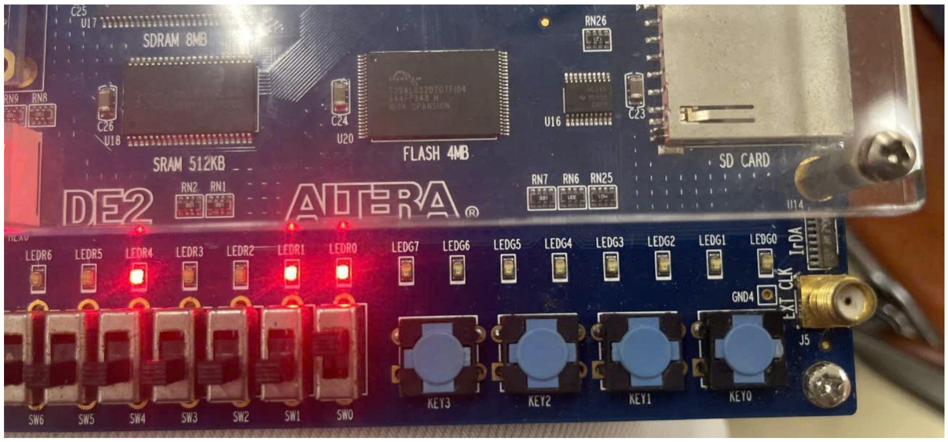
```
module sync_counter_4bit_struc (
    input  wire CLK,
    input  wire RST_N, // Reset khong dong bo, muc thap
    output wire [3:0] Q_out
);

    // 1. Tao cac day (wire) noi bo
    wire t0, t1, t2, t3; // Day cho dau vao T
    wire q0, q1, q2, q3; // Day cho dau ra Q

    // 2. Logic to hop (Combinational) de tao ra T
    // Day la mot phan cua "cau truc"

    // T0 luon = 1
    assign t0 = 1'b1;

    // T1 = Q0
    assign t1 = q0;

    // T2 = Q0 AND Q1
    and g1 (t2, q0, q1);

    // T3 = Q0 AND Q1 AND Q2
    wire t3_temp;
    and g2 (t3_temp, q0, q1);
    and g3 (t3, t3_temp, q2);

    // 3. Khoi tao (Instantiate) 4 T-FlipFlop
```

```
    // Tat ca deu dung chung CLK va RST_N

    // FF0 (Bit thap nhat)
    T_FlipFlop tff0_inst (
        .T(t0),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q0)
    );

    // FF1
    T_FlipFlop tff1_inst (
        .T(t1),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q1)
    );

    // FF2
    T_FlipFlop tff2_inst (
        .T(t2),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q2)
    );

    // FF3 (Bit cao nhat)
    T_FlipFlop tff3_inst (
        .T(t3),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q3)
    );

    // 4. Noi cac Q noi bo ra cong output
    assign Q_out = {q3, q2, q1, q0};

endmodule
```

Module T-Fliplop

```
module T_FlipFlop (
    input  wire T,     // 1 = Toggle, 0 = Hold
    input  wire CLK,
    input  wire RST_N, // Reset khong dong bo, muc thap
    output reg  Q
);

    initial Q = 1'b0;

    // Process nhay cam voi CLK va RST_N
    always @(posedge CLK or negedge RST_N) begin
        // Uu tien Reset khong dong bo (muc thap)
        if (!RST_N) begin // Giong (RST_N == 1'b0)
            Q <= 1'b0;
        end
        // Neu khong Reset, kiem tra T
```

```
        else if (T) begin // (T == 1'b1)
            Q <= ~Q; // Dao trang thai
        end
        // Neu T=0, Q se giu nguyen (Q <= Q)
    end
endmodule
```



7) Synchronous Counter n-bit VHDL behavior modeling

Testbench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

-- Entity Testbench (luon rong)
entity tb_sync_counter_nbit is
end entity tb_sync_counter_nbit;

architecture Behavioral of tb_sync_counter_nbit is

    -- 1. Dinh nghia so bit N ma chung ta muon test
    constant N_BITS_TB : integer := 8;

    -- 2. Khai bao component can test (DUT)
    component sync_counter_nbit_behav is
        generic ( N_BITS : integer := 4 ); -- Gia tri mac dinh (se bi ghi de)
        Port (
            CLK   : in  STD_LOGIC;
            RST_N : in  STD_LOGIC;
            EN    : in  STD_LOGIC;
            Q_out : out STD_LOGIC_VECTOR(N_BITS - 1 downto 0)
        );
    end component sync_counter_nbit_behav;

    -- 3. Tao cac tin hieu noi bo
    signal tb_clk   : STD_LOGIC := '0';
    signal tb_rst_n : STD_LOGIC;
    signal tb_en    : STD_LOGIC;
    signal tb_q_out : STD_LOGIC_VECTOR(N_BITS_TB - 1 downto 0);

    -- Dinh nghia chu ky clock
```

```
    constant T_CLK : time := 10 ns;

begin

    -- 4. Khoi tao DUT
    uut: sync_counter_nbit_behav
        generic map (
            N_BITS => N_BITS_TB  -- Chi dinh test ban 8-bit
        )
        port map (
            CLK   => tb_clk,
            RST_N => tb_rst_n,
            EN    => tb_en,
            Q_out => tb_q_out
        );

    -- 5. Tao Clock Process
    clk_process : process
    begin
        tb_clk <= '0';
        wait for T_CLK / 2; -- 5 ns
        tb_clk <= '1';
        wait for T_CLK / 2; -- 5 ns
    end process clk_process;

    -- 6. Tao Stimulus Process (kich thich)
    stim_process : process
    begin
        report "--- Bat dau mo phong N-bit Counter ---";

        -- Test 1: Kiem tra Reset (RST_N = 0)
        report "[Test 1] Kich hoat Reset khong dong bo";
        tb_rst_n <= '0';
        tb_en    <= '1'; -- Cho EN=1 de kiem tra Reset co uu tien hon
        wait for T_CLK * 2.5; -- Giu reset 2.5 chu ky

        -- Test 2: Nha Reset, bat dau dem (EN = 1)
        report "[Test 2] Nha Reset, bat dau dem (EN = 1)";
        tb_rst_n <= '1';
        tb_en    <= '1';
        wait for T_CLK * 5; -- Cho dem 5 chu ky (Q = 5)

        -- Test 3: Kiem tra Tam dung (EN = 0)
        report "[Test 3] Tam dung (EN = 0)";
        tb_en    <= '0';
        wait for T_CLK * 4; -- Cho 4 chu ky, Q phai giu nguyen o 5

        -- Test 4: Kiem tra Tiep tuc dem (EN = 1)
        report "[Test 4] Tiep tuc dem (EN = 1)";
        tb_en    <= '1';
        wait for T_CLK * 3; -- Dem tu 5 -> 6, 7, 8 (Q = 8)

        -- Test 5: Reset lai
        report "[Test 5] Reset lai";
        tb_rst_n <= '0';
        wait for T_CLK * 2;

        report "--- Ket thuc mo phong ---";
```

```
        wait; -- Dung mo phong
    end process stim_process;

end architecture Behavioral;
```

Top module

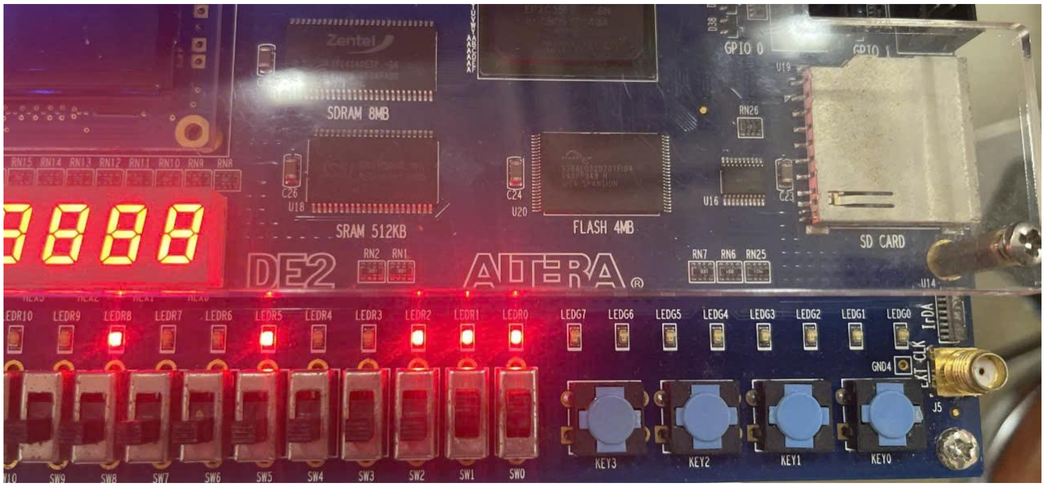
```
entity LAB2_VHDL is
    Port (
        CLOCK_50 : in  STD_LOGIC;
        SW       : in  STD_LOGIC_VECTOR(17 downto 0);
        LEDR     : out STD_LOGIC_VECTOR(17 downto 0)
    );
end entity LAB2_VHDL;

architecture Behavioral of LAB2_VHDL is

    -- 1. Khai bao component
    component clock_divider is
        Port (
            clk_in  : in  STD_LOGIC;
            clk_out : out STD_LOGIC
        );
    end component clock_divider;

    -- Khai bao component N-bit
    component sync_counter_nbit_behav is
        generic ( N_BITS : integer := 4 ); -- Gia tri mac dinh la 4
        Port (
            CLK   : in  STD_LOGIC;
            RST_N : in  STD_LOGIC; -- Reset khong dong bo, muc thap
            EN    : in  STD_LOGIC; -- Enable (1=Dem, 0=Dung)
            Q_out : out STD_LOGIC_VECTOR(N_BITS - 1 downto 0)
        );
    end component sync_counter_nbit_behav;

    -- 2. Tin hieu noi bo
    signal reset_n_signal : STD_LOGIC;
    signal enable_signal  : STD_LOGIC;
    signal q_8bit_out     : STD_LOGIC_VECTOR(7 downto 0); -- Vi du 8-bit
    signal clk_1Hz_signal : STD_LOGIC;

begin

    -- 3. Ket noi cong tac (SW)
    reset_n_signal <= SW(0); -- SW(0) la Reset (muc thap)
    enable_signal  <= SW(1); -- SW(1) la Enable (cho phep dem)

    -- 4. Ket noi den LED
    LEDR(7 downto 0) <= q_8bit_out;      -- 8 bit dem
    LEDR(8)          <= clk_1Hz_signal; -- Clock 1Hz nhap nhay
    LEDR(17 downto 9) <= SW(17 downto 9);

    -- 5a. Khoi tao Clock Divider
    clk_div_inst : clock_divider
        port map (
            clk_in  => CLOCK_50,
            clk_out => clk_1Hz_signal
        );
```

```
    -- 5b. Khoi tao Bo dem N-bit (Voi N = 8)
    uut: sync_counter_nbit_behav
        generic map (
            N_BITS => 8  -- <-- Chi dinh N = 8
        )
        port map (
            CLK   => clk_1Hz_signal, -- Cap clock 1Hz
            RST_N => reset_n_signal, -- Cap Reset
            EN    => enable_signal,  -- Cap Enable
            Q_out => q_8bit_out      -- Lay ket qua 8-bit
        );

end architecture Behavioral;
```

Module shift register and Module D fliplop

```
module clock_divider (
    input  wire clk_in,  // 50MHz
    output reg  clk_out  // 1Hz
);

    localparam MAX_COUNT = 25_000_000 - 1;
    reg [24:0] counter;

    initial begin
        clk_out = 1'b0;
        counter = 0;
    end

    always @(posedge clk_in) begin
        if (counter == MAX_COUNT) begin
            counter <= 0;
            clk_out <= ~clk_out;
        end else begin
            counter <= counter + 1;
        end
    end
endmodule

module shift_register_4bit_struc (
    input  wire CLK,
    input  wire RST_N, // Reset khong dong bo, muc thap
    input  wire D_in,  // Du lieu vao (Serial)
    output wire [3:0] Q_out  // Du lieu ra (Parallel)
);

    // 1. Tao cac day (wire) noi bo de noi cac DFF
    wire q0, q1, q2, q3;

    // 2. Khoi tao (Instantiate) 4 D-FlipFlop
    // Day la mo hinh hoa "Cau truc" (Structural)
    // D_in -> [DFF3] -> q3 -> [DFF2] -> q2 -> [DFF1] -> q1 -> [DFF0] -> q0

    // DFF3 (Bit cao nhat - Nhan D_in)
    D_FlipFlop dff3_inst (
        .D(D_in),
        .CLK(CLK),
        .RST_N(RST_N),
```

```
        .Q(q3)
    );

    // DFF2 (Nhan du lieu tu Q3)
    D_FlipFlop dff2_inst (
        .D(q3),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q2)
    );

    // DFF1 (Nhan du lieu tu Q2)
    D_FlipFlop dff1_inst (
        .D(q2),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q1)
    );

    // DFF0 (Bit thap nhat - Nhan du lieu tu Q1)
    D_FlipFlop dff0_inst (
        .D(q1),
        .CLK(CLK),
        .RST_N(RST_N),
        .Q(q0)
    );

    // 3. Noi cac Q noi bo ra cong output
    assign Q_out = {q3, q2, q1, q0};

endmodule

//----------------------------------------------
// MODULE D-FLIPFLOP (Behavioral)
// Linh kien co so de xay dung
//----------------------------------------------
module D_FlipFlop (
    input  wire D,
    input  wire CLK,
    input  wire RST_N, // Reset khong dong bo, muc thap
    output reg  Q
);

    initial Q = 1'b0;

    // Process nhay cam voi CLK va RST_N
    always @(posedge CLK or negedge RST_N) begin
        if (!RST_N) begin // Neu (RST_N == 1'b0)
            Q <= 1'b0;
        end
        else begin
            Q <= D;
        end
    end
endmodule
```



8) Right Shift Register 4-bit using Verilog structural modeling

Testbench

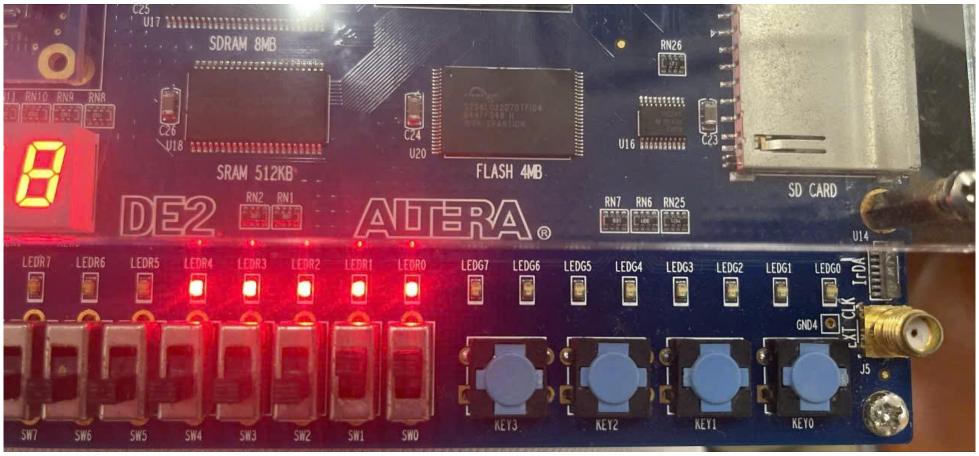
```
`timescale 1ns / 1ps

module tb_shift_register_4bit;

    // 1. Tao tin hieu
    reg  tb_clk;
    reg  tb_rst_n;
    reg  tb_d_in;
    wire [3:0] tb_q_out;

    localparam T_CLK = 10; // Chu ky clock 10ns

    // 2. Khoi tao DUT
    shift_register_4bit_struc uut (
        .CLK(tb_clk),
        .RST_N(tb_rst_n),
        .D_in(tb_d_in),
        .Q_out(tb_q_out)
    );

    // 3. Tao Clock
    initial begin
        tb_clk = 0;
        forever #(T_CLK / 2) tb_clk = ~tb_clk;
    end

    // 4. Tao Stimulus
    initial begin
        $display("Time  | RST_N | D_in | Q_out");
        $monitor("%0t | %b     | %b    | %b", $time, tb_rst_n, tb_d_in, tb_q_out);

        // Test 1: Kiem tra Reset
        tb_rst_n = 1'b0; // Kich hoat Reset
        tb_d_in  = 1'b1;
        # (T_CLK * 2);

        // Test 2: Nha Reset, nap '1'
        tb_rst_n = 1'b1; // Nha Reset
        tb_d_in  = 1'b1;
        @(posedge tb_clk); // Q_out = 1000
```

```
        // Test 3: Nap '0' va dich
        tb_d_in = 1'b0;
        @(posedge tb_clk); // Q_out = 0100

        // Test 4: Nap '1' va dich
        tb_d_in = 1'b1;
        @(posedge tb_clk); // Q_out = 1010

        // Test 5: Nap '1' va dich
        tb_d_in = 1'b1;
        @(posedge tb_clk); // Q_out = 1101

        // Test 6: Dich het
        tb_d_in = 1'b0;
        @(posedge tb_clk); // Q_out = 0110
        @(posedge tb_clk); // Q_out = 0011
        @(posedge tb_clk); // Q_out = 0001
        @(posedge tb_clk); // Q_out = 0000

        $finish;
    end

endmodule
```

Top module

```
module LAB2_VHDL (
    input  wire CLOCK_50,
    input  wire [17:0] SW,
    output wire [17:0] LEDR
);

    // 1. Tao tin hieu noi bo
    wire clk_1Hz_signal;
    wire reset_n_signal;
    wire d_in_signal;
    wire [3:0] q_4bit_out;

    // 2. Ket noi cong tac (SW)
    assign reset_n_signal = SW[0]; // SW[0] la Reset muc thap (Active-Low)
    assign d_in_signal  = SW[1]; // SW[1] la Du lieu vao (Serial In)

    // 3. Khoi tao Clock Divider
    clock_divider clk_div_inst (
        .clk_in(CLOCK_50),
        .clk_out(clk_1Hz_signal)
    );

    // 4. Khoi tao Thanh ghi dich 4-bit
    shift_register_4bit_struc uut (
        .CLK(clk_1Hz_signal),
        .RST_N(reset_n_signal),
        .D_in(d_in_signal),
        .Q_out(q_4bit_out)
    );

    // 5. Ket noi den LED
    assign LEDR[3:0] = q_4bit_out;     // 4 bit du lieu song song
```

```
    assign LEDR[4]   = clk_1Hz_signal; // Clock 1Hz nhap nhay
    assign LEDR[17:5] = SW[17:5];

endmodule
```



9) Left Shift Register 4-bit VHDL structural modeling

Testbench

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_shift_register_4bit_left is
end entity tb_shift_register_4bit_left;

architecture Behavioral of tb_shift_register_4bit_left is

    -- 1. Khai bao component
    component shift_register_4bit_left_struc is
        Port (
            CLK   : in  STD_LOGIC;
            RST_N : in  STD_LOGIC;
            D_in  : in  STD_LOGIC;
            Q_out : out STD_LOGIC_VECTOR(3 downto 0)
        );
    end component shift_register_4bit_left_struc;

    -- 2. Tao tin hieu
    signal tb_clk   : STD_LOGIC := '0';
    signal tb_rst_n : STD_LOGIC;
    signal tb_d_in  : STD_LOGIC;
    signal tb_q_out : STD_LOGIC_VECTOR(3 downto 0);

    constant T_CLK : time := 10 ns;

begin
    -- 3. Khoi tao DUT
    uut: shift_register_4bit_left_struc
        port map (
            CLK   => tb_clk,
            RST_N => tb_rst_n,
            D_in  => tb_d_in,
            Q_out => tb_q_out
        );
```

```
    -- 4. Tao Clock Process
    clk_process : process
    begin
        tb_clk <= '0';
        wait for T_CLK / 2;
        tb_clk <= '1';
        wait for T_CLK / 2;
    end process clk_process;

    -- 5. Tao Stimulus Process
    stim_process : process
    begin
        report "--- Bat dau mo phong Left Shift Register ---";
        -- In ra tieu de
        report "Time | RST_N | D_in | Q_out";

        -- Test 1: Kiem tra Reset
        tb_rst_n <= '0';
        tb_d_in  <= '1';
        wait for T_CLK * 2;

        -- Test 2: Nha Reset, nap '1'
        tb_rst_n <= '1';
        tb_d_in  <= '1';
        wait for T_CLK; -- Q = 0001

        -- Test 3: Nap '0' va dich
        tb_d_in <= '0';
        wait for T_CLK; -- Q = 0010

        -- Test 4: Nap '1' va dich
        tb_d_in <= '1';
        wait for T_CLK; -- Q = 0101

        -- Test 5: Nap '1' va dich
        tb_d_in <= '1';
        wait for T_CLK; -- Q = 1011

        -- Test 6: Dich het
        tb_d_in <= '0';
        wait for T_CLK; -- Q = 0110
        wait for T_CLK; -- Q = 1100
        wait for T_CLK; -- Q = 1000
        wait for T_CLK; -- Q = 0000

        report "--- Ket thuc mo phong ---";
        wait;
    end process stim_process;

end architecture Behavioral;
```

Top Module

```
entity LAB2_VHDL is
    Port (
        CLOCK_50 : in  STD_LOGIC;
        SW       : in  STD_LOGIC_VECTOR(17 downto 0);
```

```
        LEDR     : out STD_LOGIC_VECTOR(17 downto 0)
    );
end entity LAB2_VHDL;

architecture Behavioral of LAB2_VHDL is

    -- 1. Khai bao component
    component clock_divider is
        Port (
            clk_in  : in  STD_LOGIC;
            clk_out : out STD_LOGIC
        );
    end component clock_divider;

    component shift_register_4bit_left_struc is
        Port (
            CLK   : in  STD_LOGIC;
            RST_N : in  STD_LOGIC; -- Reset khong dong bo
            D_in  : in  STD_LOGIC; -- Du lieu vao
            Q_out : out STD_LOGIC_VECTOR(3 downto 0)
        );
    end component shift_register_4bit_left_struc;

    -- 2. Tin hieu noi bo
    signal reset_n_signal : STD_LOGIC;
    signal d_in_signal    : STD_LOGIC;
    signal q_4bit_out     : STD_LOGIC_VECTOR(3 downto 0);
    signal clk_1Hz_signal : STD_LOGIC;

begin

    -- 3. Ket noi cong tac (SW)
    reset_n_signal <= SW(0); -- SW(0) la Reset muc thap
    d_in_signal    <= SW(1); -- SW(1) la Du lieu vao (Serial In)

    -- 4. Ket noi den LED
    LEDR(3 downto 0) <= q_4bit_out;     -- 4 bit du lieu song song (Q3,Q2,Q1,Q0)
    LEDR(4)          <= clk_1Hz_signal; -- Clock 1Hz nhap nhay
    LEDR(17 downto 5) <= SW(17 downto 5);

    -- 5a. Khoi tao Clock Divider
    clk_div_inst : clock_divider
        port map (
            clk_in  => CLOCK_50,
            clk_out => clk_1Hz_signal
        );

    -- 5b. Khoi tao Thanh ghi dich TRAI
    uut: shift_register_4bit_left_struc
        port map (
            CLK   => clk_1Hz_signal, -- Cap clock 1Hz
            RST_N => reset_n_signal, -- Cap Reset
            D_in  => d_in_signal,  -- Cap Du lieu vao
            Q_out => q_4bit_out
        );

end architecture Behavioral;
```

Module Block Devider

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity clock_divider is
   Port (
      clk_in  : in  STD_LOGIC;
      clk_out : out STD_LOGIC
   );
end entity clock_divider;

architecture Behavioral of clock_divider is
   constant MAX_COUNT : integer := 24_999_999;
   signal counter     : integer range 0 to MAX_COUNT := 0;
   signal clk_1Hz_reg : std_logic := '0';
begin
   process(clk_in)
   begin
      if rising_edge(clk_in) then
         if counter = MAX_COUNT then
            counter     <= 0;
            clk_1Hz_reg <= not clk_1Hz_reg;
         else
            counter <= counter + 1;
         end if;
      end if;
   end process;
   clk_out <= clk_1Hz_reg;
end architecture Behavioral;
```

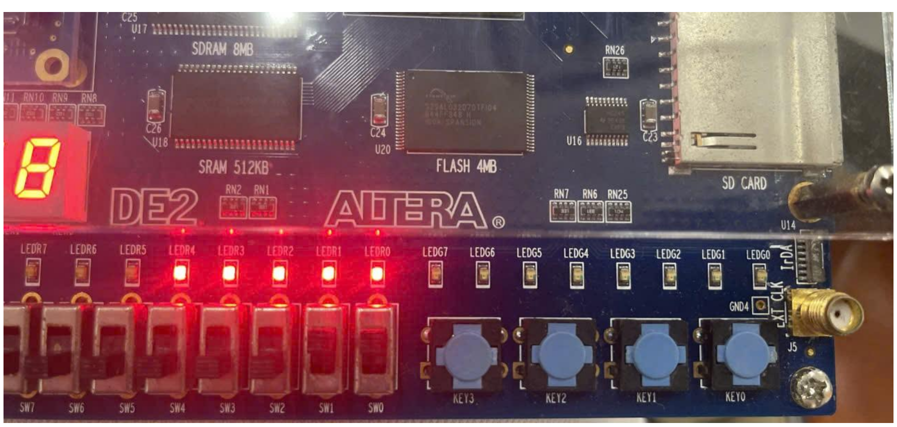
Module Shift left 4-Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_FlipFlop is
   Port (
      D     : in  STD_LOGIC;
      CLK   : in  STD_LOGIC;
      RST_N : in  STD_LOGIC; -- Reset khong dong bo, muc thap
      Q     : out STD_LOGIC
   );
end entity D_FlipFlop;

architecture Behavioral of D_FlipFlop is
   signal q_reg : STD_LOGIC := '0';
begin
   Q <= q_reg;

   process(CLK, RST_N)
   begin
      -- Uu tien Reset khong dong bo (muc thap)
      if RST_N = '0' then
         q_reg <= '0';
```

```
        -- Neu khong Reset, kiem tra canh len CLK
        elsif rising_edge(CLK) then
            q_reg <= D;
        end if;
    end process;

end architecture Behavioral;
```

Module D-Flipflop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_FlipFlop is
    Port (
        D     : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        RST_N : in  STD_LOGIC; -- Reset khong dong bo, muc thap
        Q     : out STD_LOGIC
    );
end entity D_FlipFlop;

architecture Behavioral of D_FlipFlop is
    signal q_reg : STD_LOGIC := '0';
begin
    Q <= q_reg;

    process(CLK, RST_N)
    begin
        -- Uu tien Reset khong dong bo (muc thap)
        if RST_N = '0' then
            q_reg <= '0';
        -- Neu khong Reset, kiem tra canh len CLK
        elsif rising_edge(CLK) then
            q_reg <= D;
        end if;
    end process;

end architecture Behavioral;
```



10) Universal Shift Register N-bit Verilog behavior modeling

Testbench

```
`timescale 1ns / 1ps

module tb_universal_shift_register;

   // Chon N=8 de test
   parameter N = 8;

   // 1. Tao tin hieu
   reg  tb_clk;
   reg  tb_rst_n;
   reg  [1:0] tb_sel;
   reg  [N-1:0] tb_p_in;
   reg  tb_s_in_r;
   reg  tb_s_in_l;
   wire [N-1:0] tb_q_out;

   localparam T_CLK = 10; // Chu ky clock 10ns

   // 2. Khoi tao DUT
   universal_shift_register_nbit #(
      .N_BITS(N)
   ) uut (
      .CLK(tb_clk),
      .RST_N(tb_rst_n),
      .sel(tb_sel),
      .P_in(tb_p_in),
      .S_in_R(tb_s_in_r),
      .S_in_L(tb_s_in_l),
      .Q_out(tb_q_out)
   );

   // 3. Tao Clock
   initial begin
      tb_clk = 0;
      forever #(T_CLK / 2) tb_clk = ~tb_clk;
   end

   // 4. Tao Stimulus
   initial begin
      $display("Time  | RST_N | Sel | P_in     | S_R | S_L | Q_out");
      $monitor("%0t | %b     | %b  | %b | %b   | %b   | %b",
               $time, tb_rst_n, tb_sel, tb_p_in, tb_s_in_r, tb_s_in_l, tb_q_out);

      // Test 1: Reset
      tb_rst_n = 1'b0;
      tb_sel = 2'b11;
      tb_p_in = 8'hFF;
      # (T_CLK * 2);

      // Test 2: Nha Reset (Q_out van la 00)
      tb_rst_n = 1'b1;
      # (T_CLK);

      // Test 3: Parallel Load (10100101 = 0xA5)
      tb_sel = 2'b11;
```

```
        tb_p_in = 8'hA5;
        @(posedge tb_clk); // Q_out = 10100101

        // Test 4: Hold
        tb_sel = 2'b00;
        @(posedge tb_clk); // Q_out = 10100101

        // Test 5: Shift Right (Nap 1)
        tb_sel = 2'b01;
        tb_s_in_r = 1'b1;
        @(posedge tb_clk); // Q_out = 11010010

        // Test 6: Shift Right (Nap 0)
        tb_s_in_r = 1'b0;
        @(posedge tb_clk); // Q_out = 01101001

        // Test 7: Shift Left (Nap 1)
        tb_sel = 2'b10;
        tb_s_in_l = 1'b1;
        @(posedge tb_clk); // Q_out = 11010011

        // Test 8: Shift Left (Nap 0)
        tb_s_in_l = 1'b0;
        @(posedge tb_clk); // Q_out = 10100110

        // Test 9: Reset lai
        tb_rst_n = 1'b0;
        # (T_CLK * 2);

        $finish;
    end

endmodule
```

Top module

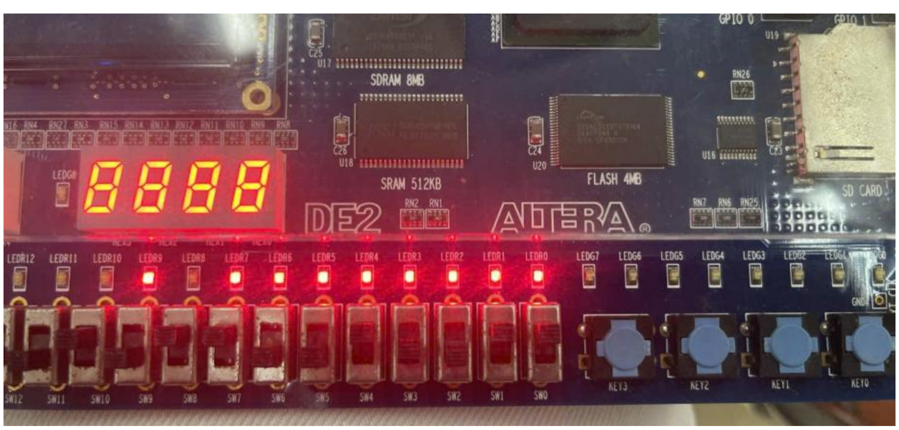
```
module LAB2 (
    input  wire CLOCK_50,
    input  wire [17:0] SW,
    output wire [17:0] LEDR
);

    localparam N = 8; // Dinh nghia so bit N = 8

    // 1. Tao tin hieu noi bo
    wire clk_1Hz_signal;
    wire reset_n_signal;
    wire [1:0] sel_signal;
    wire [N-1:0] p_in_signal;
    wire s_in_r_signal;
    wire s_in_l_signal;
    wire [N-1:0] q_out_signal;

    // 2. Ket noi cong tac (SW)
    assign reset_n_signal = SW[0];       // SW[0] = Reset (Muc thap)
    assign sel_signal     = SW[9:8];     // SW[9:8] = Chon che do
    assign p_in_signal    = SW[N-1:0];   // SW[7:0] = Du lieu nap song song
```

```
    assign s_in_r_signal  = SW[10];      // SW[10] = Vao (Dich Phai)
    assign s_in_l_signal  = SW[11];      // SW[11] = Vao (Dich Trai)

    // 3. Khoi tao Clock Divider
    clock_divider clk_div_inst (
        .clk_in(CLOCK_50),
        .clk_out(clk_1Hz_signal)
    );

    // 4. Khoi tao Thanh ghi dich da nang
    universal_shift_register_nbit #(
        .N_BITS(N) // Truyen tham so N=8 vao module
    ) uut (
        .CLK(clk_1Hz_signal),
        .RST_N(reset_n_signal),
        .sel(sel_signal),
        .P_in(p_in_signal),
        .S_in_R(s_in_r_signal),
        .S_in_L(s_in_l_signal),
        .Q_out(q_out_signal)
    );

    // 5. Ket noi den LED (DA SUA LOI)
    assign LEDR[N-1:0] = q_out_signal;     // LEDR[7:0] hien thi Q_out
    assign LEDR[8]     = clk_1Hz_signal;   // LEDR[8] hien thi clock 1Hz
    assign LEDR[10:9]  = sel_signal;       // <--- SUA DOI: Doi len LEDR[10:9]
    assign LEDR[17:11] = SW[17:11];        // <--- SUA DOI: Dieu chinh cac LED
con lai

endmodule
```

Module Shift register N-Bit

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Khong can NUMERIC_STD vi chi dich va noi bit

entity universal_shift_register_nbit is
    generic (
        N_BITS : integer := 4 -- Gia tri N mac dinh
    );
    Port (
        CLK    : in  STD_LOGIC;
        RST_N  : in  STD_LOGIC; -- Reset khong dong bo
        sel    : in  STD_LOGIC_VECTOR(1 downto 0);
        P_in   : in  STD_LOGIC_VECTOR(N_BITS - 1 downto 0);
        S_in_R : in  STD_LOGIC; -- Vao dich phai
        S_in_L : in  STD_LOGIC; -- Vao dich trai
        Q_out  : out STD_LOGIC_VECTOR(N_BITS - 1 downto 0)
    );
end entity universal_shift_register_nbit;

architecture Behavioral of universal_shift_register_nbit is

    -- Can mot thanh ghi noi bo
    signal q_reg : STD_LOGIC_VECTOR(N_BITS - 1 downto 0) := (others => '0');

begin
```

```
    -- Process mo ta hanh vi
    process(CLK, RST_N)
    begin
        -- Uu tien 1: Reset khong dong bo
        if RST_N = '0' then
            q_reg <= (others => '0');

        -- Uu tien 2: Hoat dong dong bo
        elsif rising_edge(CLK) then
            case sel is
                -- Che do 00: Hold
                when "00" =>
                    q_reg <= q_reg;

                -- Che do 01: Shift Right
                when "01" =>
                    -- Noi S_in_R vao ben trai, va lay N-1 bit ben phai
                    q_reg <= S_in_R & q_reg(N_BITS - 1 downto 1);

                -- Che do 10: Shift Left
                when "10" =>
                    -- Lay N-1 bit ben trai, va noi S_in_L vao ben phai
                    q_reg <= q_reg(N_BITS - 2 downto 0) & S_in_L;

                -- Che do 11: Parallel Load
                when "11" =>
                    q_reg <= P_in;

                -- Mac dinh
                when others =>
                    q_reg <= q_reg;
            end case;
        end if;
    end process;

    -- Gan thanh ghi noi bo ra output
    Q_out <= q_reg;

end architecture Behavioral;
```



11) Universal Shift Register N-bit VHDL behavior modeling
Testbech

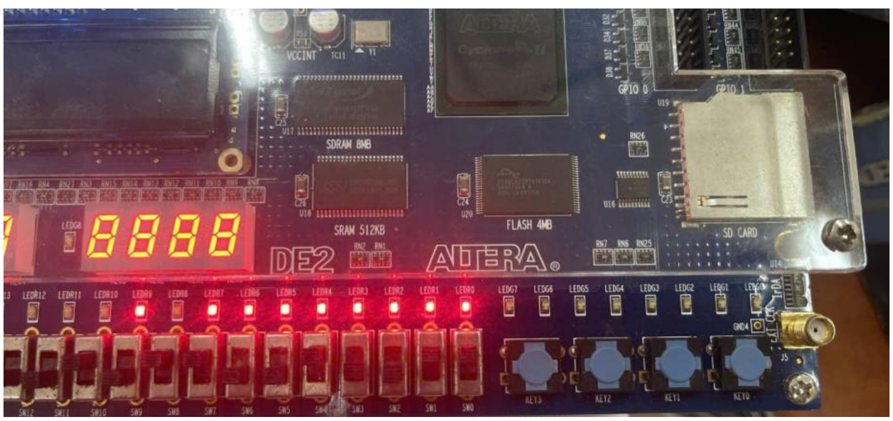
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_universal_shift_register is
end entity tb_universal_shift_register;

architecture Behavioral of tb_universal_shift_register is

    -- Dinh nghia N de test
    constant N_BITS_TB : integer := 8;

    -- 1. Khai bao component
    component universal_shift_register_nbit is
        generic ( N_BITS : integer := 4 );
        Port (
            CLK    : in  STD_LOGIC;
            RST_N  : in  STD_LOGIC;
            sel    : in  STD_LOGIC_VECTOR(1 downto 0);
            P_in   : in  STD_LOGIC_VECTOR(N_BITS - 1 downto 0);
            S_in_R : in  STD_LOGIC;
            S_in_L : in  STD_LOGIC;
            Q_out  : out STD_LOGIC_VECTOR(N_BITS - 1 downto 0)
        );
    end component universal_shift_register_nbit;

    -- 2. Tao tin hieu
    signal tb_clk   : STD_LOGIC := '0';
    signal tb_rst_n : STD_LOGIC;
    signal tb_sel   : STD_LOGIC_VECTOR(1 downto 0);
    signal tb_p_in  : STD_LOGIC_VECTOR(N_BITS_TB - 1 downto 0);
    signal tb_s_in_r: STD_LOGIC;
    signal tb_s_in_l: STD_LOGIC;
    signal tb_q_out : STD_LOGIC_VECTOR(N_BITS_TB - 1 downto 0);

    constant T_CLK : time := 10 ns;

begin
    -- 3. Khoi tao DUT
    uut: universal_shift_register_nbit
        generic map (
            N_BITS => N_BITS_TB
        )
        port map (
            CLK    => tb_clk,
            RST_N  => tb_rst_n,
            sel    => tb_sel,
            P_in   => tb_p_in,
            S_in_R => tb_s_in_r,
            S_in_L => tb_s_in_l,
            Q_out  => tb_q_out
        );

    -- 4. Tao Clock Process
    clk_process : process
    begin
        tb_clk <= '0';
        wait for T_CLK / 2;
        tb_clk <= '1';
```

```
        wait for T_CLK / 2;
    end process clk_process;

    -- 5. Tao Stimulus Process
    stim_process : process
    begin
        report "--- Bat dau mo phong Universal Shift Register ---";
        report "Time | RST | Sel | P_in     | S_R | S_L | Q_out";

        -- Test 1: Reset
        tb_rst_n <= '0';
        tb_sel   <= "11";
        tb_p_in  <= (others => '1');
        wait for T_CLK * 2;

        -- Test 2: Nha Reset
        tb_rst_n <= '1';
        wait for T_CLK;

        -- Test 3: Parallel Load (10100101 = x"A5")
        tb_sel   <= "11";
        tb_p_in  <= x"A5";
        wait for T_CLK; -- Q_out = 10100101

        -- Test 4: Hold
        tb_sel <= "00";
        wait for T_CLK; -- Q_out = 10100101

        -- Test 5: Shift Right (Nap 1)
        tb_sel   <= "01";
        tb_s_in_r <= '1';
        wait for T_CLK; -- Q_out = 11010010

        -- Test 6: Shift Right (Nap 0)
        tb_s_in_r <= '0';
        wait for T_CLK; -- Q_out = 01101001

        -- Test 7: Shift Left (Nap 1)
        tb_sel   <= "10";
        tb_s_in_l <= '1';
        wait for T_CLK; -- Q_out = 11010011

        -- Test 8: Shift Left (Nap 0)
        tb_s_in_l <= '0';
        wait for T_CLK; -- Q_out = 10100110

        -- Test 9: Reset lai
        tb_rst_n <= '0';
        wait for T_CLK * 2;

        report "--- Ket thuc mo phong ---";
        wait;
    end process stim_process;

end architecture Behavioral;
```

Top module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

------------------------------------------------
-- MODULE TOP-LEVEL (Ten file: LAB2_VHDL.vhd)
------------------------------------------------
entity LAB2_VHDL is
  Port (
    CLOCK_50 : in  STD_LOGIC;
    SW       : in  STD_LOGIC_VECTOR(17 downto 0);
    LEDR     : out STD_LOGIC_VECTOR(17 downto 0)
  );
end entity LAB2_VHDL;

architecture Behavioral of LAB2_VHDL is

  -- Dinh nghia so bit N
  constant N_BITS_TOP : integer := 8;

  -- 1. Khai bao component
  component clock_divider is
    Port (
      clk_in  : in  STD_LOGIC;
      clk_out : out STD_LOGIC
    );
  end component clock_divider;

  component universal_shift_register_nbit is
    generic ( N_BITS : integer := 4 ); -- Gia tri mac dinh
    Port (
      CLK    : in  STD_LOGIC;
      RST_N  : in  STD_LOGIC; -- Reset khong dong bo
      sel    : in  STD_LOGIC_VECTOR(1 downto 0);
      P_in   : in  STD_LOGIC_VECTOR(N_BITS - 1 downto 0);
      S_in_R : in  STD_LOGIC; -- Vao dich phai
      S_in_L : in  STD_LOGIC; -- Vao dich trai
      Q_out  : out STD_LOGIC_VECTOR(N_BITS - 1 downto 0)
    );
  end component universal_shift_register_nbit;

  -- 2. Tin hieu noi bo
  signal clk_1Hz_signal : STD_LOGIC;
  signal reset_n_signal : STD_LOGIC;
  signal sel_signal     : STD_LOGIC_VECTOR(1 downto 0);
  signal p_in_signal    : STD_LOGIC_VECTOR(N_BITS_TOP - 1 downto 0);
  signal s_in_r_signal  : STD_LOGIC;
  signal s_in_l_signal  : STD_LOGIC;
  signal q_out_signal   : STD_LOGIC_VECTOR(N_BITS_TOP - 1 downto 0);

begin

  -- 3. Ket noi cong tac (SW)
  reset_n_signal <= SW(0);
  sel_signal     <= SW(9 downto 8);      -- SW[9:8] = Chon che do
  p_in_signal    <= SW(N_BITS_TOP - 1 downto 0); -- SW[7:0] = Du lieu nap song song
  s_in_r_signal  <= SW(10);              -- SW[10] = Vao (Dich Phai)
  s_in_l_signal  <= SW(11);              -- SW[11] = Vao (Dich Trai)
```

```
    -- 4. Khoi tao Clock Divider
    clk_div_inst : clock_divider
        port map (
            clk_in  => CLOCK_50,
            clk_out => clk_1Hz_signal
        );

    -- 5. Khoi tao Thanh ghi dich da nang (Voi N = 8)
    uut: universal_shift_register_nbit
        generic map (
            N_BITS => N_BITS_TOP  -- Chi dinh N = 8
        )
        port map (
            CLK    => clk_1Hz_signal,
            RST_N  => reset_n_signal,
            sel    => sel_signal,
            P_in   => p_in_signal,
            S_in_R => s_in_r_signal,
            S_in_L => s_in_l_signal,
            Q_out  => q_out_signal
        );

    -- 6. Ket noi den LED (Da sua loi trung lap)
    LEDR(N_BITS_TOP - 1 downto 0) <= q_out_signal;   -- LEDR[7:0]
    LEDR(8)                       <= clk_1Hz_signal;  -- LEDR[8]
    LEDR(10 downto 9)             <= sel_signal;      -- LEDR[10:9]
    LEDR(17 downto 11)            <= SW(17 downto 11);

end architecture Behavioral;
```



For each circuit, Do the following steps:
Step 1 : Draw the Schematic of this circuit
(Show Schematic in Lab report)
Step 2 : Write the truth Table for this circuit
(Show The Truth Table in Lab report)
Step 3 : Write the Verilog Module to implement this circuit (using structural, data flow, and behavior modeling)
(Show VHDL codes or Verilog Codes of this module in Lab report)

Step 4 : Write the testbench to simulate the Verilog modules of this circuit

(Show simulation results in Lab report)

Step 5 : Write the Top-level Verilog Code to implement the Verilog modules of this circuit in DE2-FPGA Kit

(Show implementation results in Lab report)

Step 6 : Upload your study evidences in Your Github Account

(Show your Github link)

IV. LAB REPORT GUIDELINES

Students write up a report on the Verilog and VHDL implementation experiment projects created in this lab. The lab report should include Circuit Schematics, Verilog Module Codes, Verilog test bench codes, Top level module to implement the required circuit in FPGA KIT and evidences of data output evidences to validate the experiments (The Captured Screens, Photo of FPGA Kit implementation results).